Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Mechanical Engineering Department

Initial Design Idea

Embedded motion control
4SC020

Quartile 4: 2015 - 2016

Josten, T.J. 0811028
Leenen, T.C.P.F 0790737
Plantinga, M. 0832751
Reinders, J.M.F. 0816951

Eindhoven, April 27, 2016

Introduction

The goal of this project is to make a robot (PICO or TACO) navigate autonomously and as seamlessly as possible through a maze and find the exit. The robot has a computer integrated in it that runs on Linux, with ROS (Robot Operating System) running on top. The software has to be written in the `C++` programming language. To achieve the goal, a software architecture has to be designed that structurally makes room for the different requirements, functions, components, specifications and interfaces. In this report the initial software idea will be discussed on the basis of these five criteria.

Requirements

To achieve the final goal several requirements are determined, which can be seen in the following list.

- Navigate autonomously to the exit of the maze as fast as possible
  cruise as fast as possible while maintaining the different abilities
- Avoid obstacles
  recognize the different obstacles (e.g. walls) and keep a "safe" distance from them
- Avoid getting trapped in a loop of the maze
  recognize if the robot is navigating through the same path over and over and exit the loop
- Create a map of the maze
- Recognize door, open it and drive through it
- Navigate in open spaces
  navigate if no obstacle is in sight
- Scalable system
  the software should be able to work independently of the size and structure of the maze

Functions

The functions can be divided into two groups: the basic functions and the skill functions. The basic functions are basic actions that the robot will do. The skill functions are a set of actions to accomplish a certain goal.

The basic functions consist of:

- Actuation of the robot:
  - Provide signals to the actuators of the robot. Realize the desired motion by using a controller and meeting the requirements.
  - Input: location of robot, Output: motion of robot
- Detect:
  - Characterize different types of corridors based on telemetry provided by the Laser Range Finder (LRF).
  - Input: measured x, y and theta; Output: type of corridor

The skill functions consist of:

- Mapping:
  - Create and update a map of the explored maze. The robot will recall this map as its future moves will depend on this map.

  - Input: current x, y and theta of the robot; Output: new/updated maze map, new/adjusted objective of the robot.

- Feedback:
  - Check position of robot with position of the created map. Prevent the robot from collisions with the walls or other obstacles.
  - Input: current x, y, theta, LFR data and objective; Output: motion of the robot.
- Decision:
  - Determine the following action based upon the current location of the robot
  - Input: current x, y, theta and position in the map; Output: motion of the robot
- Monitor:
  - Control the exploration of the maze and prevent the robot from getting stuck in a loop
  - Input: current x, y and theta of the robot; Output: previously unexplored area
- Door check:
  - Wait at the potential door location for a predetermined time period while scanning the distance to the potential door to check if it opens
  - Input: current x, y and theta of the robot; Output: A door that either opens or stays closed, followed by pico's new objective based upon the result.
- Obstacle check:
  - Measure the preset minimum safe distance from the walls or measure not moving as expected according to the driving action.
  - Input: current and expected x, y and theta of the robot; Output: new motion of the robot based upon the result.

Components

To fulfil the previously mentioned functions and achieve the goal, the software should contain different components that interact with each other. This can be seen in Figure 1.

The `C++` code should contain the components shown in Figure 1. There should be a task manager to switch between different tasks. An algorithm should be made that decides which task is performed or whether different tasks are performed simultaneously.

An algorithm should be implemented for controlling the robot and accurately position it, this algorithm uses the environment model, which is made using the laser range finder and omni-wheel encoders. This world model is logged and will be continuously updated during the maze solving of the robot.

The robot should have several skills and these should be programmed effectively with a fast algorithm. For the world model, the robot needs a mapping skill and it needs to determine its position in this world model. To solve the maze the robot needs a trajectory planning which uses an efficient maze solving algorithm (e.g. Trémaux).

Eventually when the robot has solved the maze, the algorithm has to be stopped.

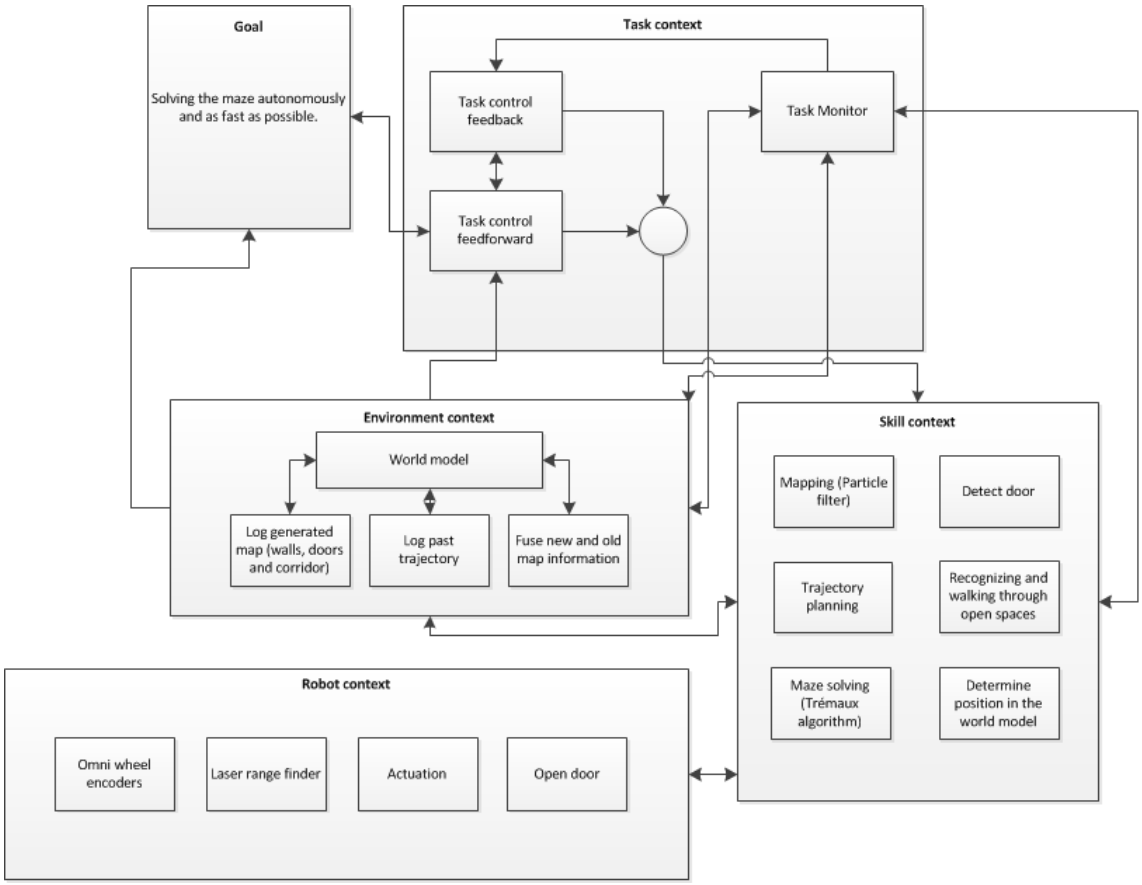

Figure 1: Structure of the software implementation

Specifications

In the specifications the tasks that the robot has to conduct are quantified (i.e. given a value).

- The maximal translational velocity of PICO is 0.5 m/s
- The maximal rotational speed of PICO is 1.2 rad/s
- PICO has to solve the corridor challenge within 5 minutes
- PICO has to solve the maze challenge within 7 minutes
- PICO may not stop moving for more than 30 seconds (e.g. when opening the door)

Interfaces

To interpret the data that the robot collects, it is convenient to make a graphical user interface that visualizes this data. The omni-wheel encoder in combination with the laser range finder can be used to visualize the path of the robot and make the world model. The encoder data should be transformed to robot coordinates with a kinematic model of the robot.

The laser range finder also produces data, and to see if this data is interpreted in the right way a visualization should be made. It can be used to see if walls, doors and exits are detected in the right way. Possible algorithms for this are the Hough transform and particle filter.